21
The HELP System Development Tools

T. Allan Pryor, Homer R. Warner,
Reed M. Gardner, Paul D. Clayton,
and Peter J. Haug

The HELP (Health Evaluation Through Logical Processing) system, developed at the LDS Hospital and University of Utah in Salt Lake City, is a complete hospital information system (HIS) that supports not only traditional HIS functions, but routinely implements decision-making applications for clinical care.* The following list enumerates the clinical applications that are currently available on the HELP system:

Admit/Discharge/Transfer
Order Entry/Charge Capture
Medical Records
Intensive-Care Unit Automation
Nursing Charting/Care Plans/Acuity
Radiology
Surgery Scheduling/Management
Laboratory Integration/Pathology
Microbiology Alerting/Infectious Disease
Pharmacy
Laboratory Alerting
Quality of Care
Blood Gas Analysis
Pulmonary Function Laboratory
Physician Remote Access
Respiratory Care
Cardiology–ECG, Imaging, Catheterization Laboratory
Obstetrics/Delivery
Patient History

HELP is designed to facilitate the use of medical computing technology in increasingly new areas without requiring the application developer to use low-

*For a description of HELP, see Blum BI (ed): Information Systems for Patient Care: New York, Springer-Verlag, 1984, chap 8.

level programming languages. Five major software tools have been written to allow clinicians, who may not have a knowledge of programming, to create or modify applications:

1. PTXT (pointer to text) data dictionary for the definition of data elements
2. GQAP (General Questionnaire Asking Program) for the definition of data-entry screens
3. HCOM (Help compiler) for the definition of decision logic
4. PAL (PTXT Application Language) for the definition of formatted reports
5. STRATO for the creation and analysis of research populations

This chapter discusses the design and use of these HELP system software tools.

PTXT Data Dictionary

In order for information in the clinical data base (facts about the patient) to be usable, data must be stored in some uniquely identifiable form (as opposed to free text). Each medical term used must have been previously defined in a data dictionary. Such an authoritative standard ensures a common terminology for all specific items of patient data. In the HELP system, all clinical data are physically stored on disk in a coded format. The code for each term is derived from the position of the defined term within a hierarchical structure of medical terminology. The coded format facilitates rapid, selective retrieval of data from the data base and allows the use of hierarchical relationships. The codes are *retranslated* to common medical terminology for data review.

In the data dictionary a relationship exists between each entity defined by one of several "fields." The first field is the hierarchical code that is used as a unique key for the entity. The medical terminology (text) is stored in the second field. The dictionary is called PTXT because of this fundamental relationship. Additional fields contain information such as hospital charges, hospital costs, and financial account numbers. The final field consists of key words associated with the term.

The key words are used to reference data elements previously defined in the data dictionary. For example, if one required a rule about the presence or absence of aspirin, one would enter the keyword "ASA" (acetylsalicylic acid), and all entries in the data dictionary for which this keyword was entered as an identifier would be displayed. In the case of "ASA," the dictionary currently has six entries, ranging from actual drugs to suggestions about wound management or home medication history questions. The dictionary thus gives every possible usage of aspirin that has been entered into the system and the user can select those appropriate. The user makes this selection by picking from the list of terms that are defined for the keyword entry. Thus, it is not necessary for the user to actually know the exact codes for the terms to properly use those terms in the application under development.

The procedure (service costs and account code) field of dictionary record is used for those items that represent chargeable/billable procedures or services. This allows the system to automatically capture charges as the clinical results are stored in the patient's file.

The structure of the PTXT dictionary is hierarchical. The names of each level of the hierarchy (starting at the top) include data class, data type, field code, noun, adjective, adverb. The names do not correspond exactly to equivalents in English grammar—only to levels of the hierarchy. In addition to terms in the hierarchical structure, one can also define terms that act as modifiers. These modifiers can apply to multiple terms within a hierarchy.

Data classes are used to define individual subspecialty areas of medicine. For example, drugs comprise one data class; ECG results or measurements are in another. Other data classes include surgical procedures, radiology procedures, chemistry laboratory, microbiology, hemodynamics, etc.

Data types identify data strings with different structures. The system supports "type zero" strings whose structures are defined relations, and the location of the data defines its meaning (which byte and bit of the string). This structure is equivalent to the logical model in relational data bases.

For "type one" data, the content of the string is self-defining. Certain symbols tell what kind of item will occur next in the string. For example, a delimiter is used to show that a modifier, a noun, or a 32-bit floating point number will follow; following that information another symbol defines the next element of the string. This type of format allows very flexible construction of data strings from questionnaires in which the number and types (numeric data, multiple-choice options, etc) of answers vary from patient to patient. Over the years, most of our data strings have migrated from the structured relations found in "type zero," to the more flexible format of the "type one" string. This migration has occurred because application developers felt that there was more versatility, efficiency, and convenience in the latter approach.

"Type three" strings are used to store the results of a computer-generated decision. Therefore, type-three data consist of all the alerts, data interpretations, treatment suggestions, and diagnoses that have been made by the system. Through a data-driven mechanism, the storage of type-three data may activate subsequent decision frames whose logic criteria reference the stored results of the original decision frame. In the philosophical sense, this method of storing data is similar to (what some refer to as) the blackboard method for control: the results of evaluating one "chunk" of knowledge are written on the blackboard (patient data base) so that other rules can be activated.

The final type of string "(type seven)" is reserved for a transaction log. Type-seven strings have a unique, sequentially ascending key for each day and each ancillary service department to ensure that every transaction is logged (for management and financial purposes). The content of these strings may also be referenced in the decision-making logic or reporting logic of the system, as well as for routine hospital administrative and financial functions.

```
RELATION cbc;
BEGIN
ITEM time          STRINGTIME;
ITEM wbc           CODE(15 1 62 1 1);
ITEM rbc           CODE(15 1 62 1 2);
ITEM hgb           CODE(15 1 62 1 3);
ITEM hct           CODE(15 1 62 1 4);
End;
```

Figure 21.1. A PAL relation showing PTXT codes for specifying complete blood count (cbc) (rbc, red blood cells; wbc, white blood cells; hgb, hemoglobin; hct, hematocrit).

Field codes are the classification level below data type in the PTXT hierarchy (see Figure 21.1). For example, in the department of radiology, chest examinations comprise one field code, nuclear medicine another, and gastrointestinal examinations a third. In all, there are 14 field codes to cover the variety of examinations that may be performed (angiograms, computed tomography, magnetic resonance imaging, etc). Another series of field codes refer to the general categories of radiology findings. In pharmacy, the different field codes refer to the classifications of drugs—heart medications, analgesics, antipyretics, antibiotics, etc.

Nouns, adjectives, and adverbs are the next levels of classification when type-one strings are used. Within the radiology data class, nouns refer to specific procedures (two view posterior-anterior and lateral, one view anterior-posterior portable, ribs 4 views, etc). There are no lower levels (adjectives and adverbs) in the case of examinations. However, radiology findings are reported as nouns in a different field code within the radiology data class, and there are further definitions in the hierarchy below this level, eg, adjectives tell whether the infiltrate is alveolar or interstitial, and adverbs describe the pattern of the infiltrate. In the pharmacy data class, however, there are adjectives and adverbs. The noun in the pharmacy application is a further subclassification of drugs, eg, salicylates; adjectives refer to the specific generic or widely recognized brand name of the drug, eg, aspirin; and the adverbs give the dosage and form of the drug, eg, 325-mg tablet. Modifiers are defined when they apply to more than one term in the hierarchy, ie, the route of drug administration (oral, intravenous, intramuscular, etc) can apply to all drugs. The modifying, nonhierarchical terms "left" and "right" can apply to arms, ankles, and mammograms in radiology.

The system user sees the data dictionary from several perspectives. The first-time user who is entering terminology that has not yet been defined in the dictionary must determine the hierarchy for the domain. This entails many of the problems associated with choosing a knowledge representation for a problem. Because one of the main functions of the codes is to facilitate rapid retrieval of data, it is important that these developers foresee the ways in which the data will customarily be asked for and retrieved. The user must then enter the text, codes, and keywords/synonyms using the dictionary editor.

For the person who is using terminology that has already been defined, the task is much simpler. The editors for the data acquisition and report functions, as well as the decision-making logic, allow the entry of keywords. The user never sees the codes, but has an unambiguous selection of ways in which the terms have been previously used in the system.

There are two principal reasons why the hierarchical structure is advantageous. First, the reporting or decision-making logic can be written much more efficiently. By specifying "heart medication," one does not have to search for digitalis or beta blockers by all the possible brand names. This hierarchical structure is equivalent to the . . . IS A . . . or . . . A KIND OF. . . rules that exist in rule-based or semantic-net approaches. Because these relationships are implicit in the structure, far fewer rules are needed. Second, the data retrieval for decision-making logic is more efficient. All the data in one data class are stored together in the patient file. Hierarchical keys ensure that data will be retrieved with fewer disk accesses than would be the case if the terms were listed as ASCII (American Standard Code for Information Interchange) symbols, with the primary organization being dependent upon the alphabetical spelling of the term.

Set against the advantages for retrieval and management are several disadvantages of the system. The first is that when patient data are displayed, the facts about a patient must be translated into a text-based representation. This translation requires multiple-disk accesses to the dictionary, especially if one is generating a comprehensive report. The impact of this bottleneck is appreciated when one realizes that at LDS Hospital, for instance, we have 520 printers and terminals attached to the system. Many users of these devices will be simultaneously seeking to display data. We have solved this problem to some extent by distributing the dictionary file among multiple disks.

A more intrinsic problem occurs when one wishes to combine data that cross the hierarchical structure of the dictionary. For example, the radiology applications may address fractures of a particular bone while the surgery applications need to address surgical reduction of the same bone. This problem has been resolved in the past by the definition of redundant terms. We are not satisfied with this approach as the breadth of our knowledge and clinical data bases expand; therefore, we have developed data strings that contain elements from different data classes.

The third problem is not generic to the PTXT system, but is caused by multiple users who often define the terminology from the limited perspectives of their immediate domain and do not consider alternative uses and hierarchical structures. There needs to be some authoritative terminology of the content of the dictionary. We have relied on manual methods in the past and are now embarking on automated methods of management, which will hopefully preserve the freedom that currently exists.

The final disadvantage of the system is that once data have been stored in the patient archival records, using a specific set of codes defined in the dictionary, those dictionary entries may not be changed because such change precludes the use of all past archival data. As in the previous instance, this limitation is

not predicated on the hierarchical nature of our dictionary, but is a universal problem.

In summary, the data dictionary has developed tremendous flexibility as applications developers and logic authors have contributed to the content of the comprehensive system. We currently have about 114,878 terms defined in the dictionary. The coded format provides efficient data retrieval for the decision-making aspects of the system. Codes are hidden from clinical users, who see only the hierarchical relations that have previously been defined and may not be easily altered once data have been acquired and stored.

GQAP–The Screen Entry Software Tool

Two general system tools are available in the HELP system to facilitate the acquisition of data entered from terminals by the clinical staff. The first is a generalized screen menu and utilization tool called GQAP. The second is the DDA (Decision-Driven Data Acquisition) facility incorporated as a part of the decision support system of HELP; this support system is explained in the section on decision support tools of HELP.

The GQAP system provides the nonprogrammer with a screen editor that defines data-entry questionnaire. The questionnaire has three functions: (1) linking the entered data to PTXT, (2) providing error checking of the entered data, and (3) providing user-defined logic for control of the screen presentation.

No original formatting of the screens is permitted; that is, a default format is provided and required of the user once the type of screen is defined. Five screen types are available. The first is the data-entry type and provides a fixed set of data fields to be entered. After the data screen is displayed, the user is prompted to enter data in each field presented. The user may choose not to enter data in a prompted field by pressing the return key without entering a data value, but in all cases each field is presented to the user for some action. The data-entry screens are used primarily for entry of fixed sets of medical data, such as vital signs, which are usually measured and entered simultaneously.

The second screen type is the multiple-choice screen, which presents a list of possible fields. In defining a multiple-choice field, the screen programmer will define the number of choices that the user may make. After an entry is selected, a value may also be entered for the field. In fact, if the field is declared a numeric field, a numeric value must be entered for that choice. If no numeric values are entered, the system prompts the user for entry of the numeric before continuing to the next screen. All entry choices and values are displayed on a command line at the bottom of the screen. An example of the use of multiple-choice screens is in the creation of nursing care plans. The screen lists possible actions that may be performed on a patient, and the nurse chooses only those applicable to each case.

The third type is a general screen that allows the user to enter either a PTXT code or a keyword that will display the appropriate PTXT code. This screen type

is valuable when the number of selections is prohibitively large, and the entry of a code would prove simpler and more efficient.

The fourth screen type is the free-text-entry screen. These screens are used to enter a variable-length text string into the patient's record.

The fifth general screen type supported in GQAP is the time tag, used to set the chronological time associated with the data stored in the patient's file. In programming a time screen, the developer may choose to have the time entered by the user or default to the present time or default to the last time entered by the user.

Common to the definition of data-entry screens and multiple screens are the header, the fields, the diagnostic logic, and the follow-up logic. The header is a prompt (message) that is displayed at the top of the screen. Its primary function is to instruct the user on how to enter data. The definition of the fields of the screen is a major requisite as the developer determines the text to be displayed on the terminal, the associated PTXT codes, and special characteristics of the field. When defining the text of the field, the developer may enter the text directly, or use the text defined for the PTXT code, if the field is to be associated with a PTXT code. If the field is to be associated with a PTXT code, then the developer can either enter the code directly, or select from lists generated by the keywords chosen from the data dictionary. The characteristics for the field include the format of the value that must be entered; the requirement to store the PTXT code in the patient's data base; the need to store the PTXT code in special buffers for transmission to other processes on the computer; the hidden field characteristic that causes this field not to be displayed when the screen is displayed, but forces the field to be chosen if the screen is displayed; and the billing state of the field if it is a chargeable item.

Diagnostic logic may be included with each screen. For data-entry screens, the diagnostic logic can perform limit checks on the entered data. In declaring the diagnostic logic, the screen developer can define both the diagnostic statement to be presented if the diagnostic is true and a Boolean logic statement, which, if satisfied, will cause the diagnostic statement to be displayed on the screen. For example, a diagnostic statement for an input field for entering the patient's age could be written in the question as "Print STRANGE AGE if 1 > C110 OR 1 < C5. In this example the field "AGE" is assumed as the first field in the question. The diagnostic may allow the user to override the logic and continue to the next question. With multiple-choice screens the diagnostics are used not only to check the values of the entries, but also to validate the consistency of the choices themselves. That is, if the screen presents more than one choice set (one or more fields in a multiple-choice screen) where only one selection from each set is logical, a diagnostic can be provided which ensures that only one choice from each set is made. Likewise, if a selection from a choice set is required, the diagnostic can ensure that the next question is not presented until a choice from that set is made. This is often used when the questionnaire developer wishes to combine two questions on a single screen, rather than use two screens for the presentation. The developer is not limited to a single diagnostic statement per question, but may include as many as are appropriate.

Follow-up logic may also be included in the formatting of the screen. As with the diagnostic logic, many follow-up statements may be included. If more that one statement is included and satisfied, then both will eventually be presented to the user. The general format of a follow-up statement is "ASK (question number) if (Boolean statement)." When developing an input screen, the programmer declares an initial question, or set of questions, as key questions and immediately places them on the "screens to be displayed" stack when the questionnaire is activated. Presentation of successive screens is then accomplished by "popping" the "screens to be displayed" stack and presenting the screen at the top of the stack. The follow-up logic activates more screens to be added to the stack, and the presentation of each screen causes it to be removed from the stack. When the stack is empty, the questionnaire is terminated and control returned to HELP system for initiation of other applications.

Options that control the presentation and storage of the acquired data before it is stored in the patient file are available. Since the PTXT data base is string-oriented, the developer must decide on the precise moment when the entered data should be packed into a PTXT string and stored in the patient's record. This is normally done by setting a question flag, which will cause all data in the patient store buffer to be packed and stored and the buffer to be reset for capture of additional data.

Because some questionnaires are used very frequently, a command string option is provided to allow for more rapid entry of data. Using a command string, the user may preanswer screens before their presentation. For example, when ordering an ECG, the ward clerk may know that a routine 12-lead ECG for tomorrow morning is selection 3 on the first screen, followed by selection 4 on the next screen, with the final selection being 2 on the last screen. In such an example, the clerk could then enter the appropriate command string (3;4;2;) and have the order stored without having to see the actual screens.

The use of GQAP as a data-entry tool has proved extremely useful. Among the applications where the data entry is entered using GQAP questionnaire are order entry, nurse charting, nurse care plans, research data entry, and quality-control data entry.

HCOM—the Decision Support Software Tool

In the HELP system, knowledge is represented in the form of frames known as HELP sectors. Each of these frames has the following components:

1. text string or title
2. list of destinations to which the text is to be sent
3. list of data items from the HELP dictionary that are to be used in making the decision
4. logical expressions that define the decision criteria.

In this section we describe the editor (HCOM) and the syntax used to build and maintain HELP sectors and to express their content in a form understandable to clinicians and students. The data items in a sector must be explicitly defined in the PTXT file, (the dictionary) of the HELP system. If an appropriate term cannot be found, it must be added to PTXT before it can be used by HCOM to construct a decision frame. The HELP sector text, or message itself, becomes part of the PTXT file.

HCOM offers several modes of entry to the user who is building or modifying a sector. For the novice user, a set of menus that displays the options available at each point in the process is presented. For the more experienced person these commands may be entered as a command string of sequential operations to be performed, and HCOM checks that only allowable commands are requested at each step. To illustrate how HCOM is used, consider the following example. To begin, one may enter a '?', and HCOM displays the menu of available verbs:

```
O open, L list, LE list and explain, P print, A add, I insert,
D delete, R replace, F fix
```

Since sectors are organized into blocks for purposes of both control and efficiency, the first action by HCOM must be to open the block of sectors to be edited. After entering "O" followed by the block number, one is ready to create a new sector by entering "A" followed by "return." The next menu is a set of objects appropriate for the verb "A".

```
B block of sectors, S sector, ST sector text, STM sector text modifier, I item,
FE final evaluation
```

In this example, a new sector is created by entering "S," and HCOM will assign a new sector number. The sector text is usually entered first. The operator enters "AST" and is prompted to enter the sector text enclosed in quotation marks, followed by one or more keywords, each in quotes. These keywords allow retrieval of this sector from the PTXT file should it be referred to by the editor in some other sector.

The message generated by a sector is controlled by the sector logic, which makes use of sector text modifiers. As a simple example, consider the sector text, "This patient's history suggests pneumonia, associated with =, =, =." If one now adds sector text modifiers "cough," "fever," "chills," following the command string "ASTM," one or more of these words can be made to print out a message conditional upon the logic in the sector and the data in the patient's file.

Each HELP sector is organized into a list of alphabetically labeled items. Each item may specify that a search is to be performed on the patient's data file (search item), an arithmetic or logical relation among preceding items to be evaluated (arith item), or a probabilistic expression using sensitivity and specificity of an observation (prob item). HCOM provides special features for the creation of each of these item types.

The command string "AIS?" will initiate the creation of a search item by displaying a menu of the classes of data in PTXT, such as hematology, history, and physical examination, ECG, blood gas, and drugs. The data class is selected from the menu by entering an index number followed by one or more keywords to identify the desired text string from PTXT (which currently has 114,878 terms). One may select from the list displayed, or enter a new set of keywords (or word parts), until the desired text is displayed. At this point HCOM allows a set of five constraints that further defines the data appropriate for the decision to be made:

1. *FROM.* Since a patient's file may contain more than one record of any given element defined in the PTXT (ie, white blood count), a time constraint may be created for the search. For example, a variable may only be relevant if it occurs "from 24 hours before now" or "from 5 minutes after the last dose of antibiotic searched for in item C."
2. *TO.* An upper time boundary is also placed on each search. This activates an elaborate sequence of events before the criteria for a decision can be fulfilled. The default-time criteria is "from 48 hours ago" "to now" if not explicitly defined.
3. *MOD.* Even within a selected time interval more than one instance of the specified PTXT item may have been recorded for a given patient. One of the following modifiers may be used to specify the desired information: *first, last, maximum, minimum, frequency, nearest* in time to some other item, *average, mode*, and *trend*. "Last" is the default, since most decisions use the latest value for a variable.
4. *IF.* Conditional statements may be used to select only certain qualifying fields. Since each search item may specify more than one subitem to be sought, logical relationships among these subitems may be defined with the IF clause. The default logic is the union of all subitems.
5. *USE.* Three parameters are generated for each search item: existence, value, and time. The "USE" clause allows assignment of any value to the search item for later reference in a subsequent logic statement.

Arithmetic items may use any mathematical function needed to represent the logical model of a decision. An arithmetic item may be used to derive a value assigned to that item, or any other item label, for future reference, or to perform conditional termination or branching ahead in the sector. For example, one can assign weights to reflect parameters such as "frequency" and "evoking strength" to model systems such as INTERNIST-1.

Bayesean probability may be implemented directly with HCOM using the PROB item. If a PROB item is chosen, four parameters are possible:

1. the item whose value or existence is to be used (The item can be continuous or binary in nature and can be obtained directly from a search item or derived through an arithmetic item.)
2. the probability of the decision being true before consideration of the above item
3. the sensitivity
4. the specificity in the context of the decision represented by this sector. The calculated posterior probability can be used as the apriori probability in a subsequent PROB item.

A set of final evaluations, using arithmetic operators, can manipulate any items (value, existence, or time) in the sector. The values statement can be inserted into a position occupied by an equal sign in the sector text message, or as an index to select a sector text modifier.

If a sector is true and is stored in a patient's record, it may then be accessed by a search item in another sector. If a search for an evaluated sector is unsuccessful, that sector will be run and the results returned to the search item in the original sector (nesting).

One may, through HCOM, specify which search items in a sector are to be used in connection with the clinical care of a patient. In addition, a search item may be flagged to allow the sector to differentiate the absence of an item in a search from the absence of its "parent" in the PTXT hierarchy (ie, if the patient said he did not have a cough, HELP will infer a "no" answer to the question, "Do you have a cough productive of blood-tinged sputum?").

HCOM allows the user to specify where the text of a sector should be directed if the sector comes true. This is done by choosing one, or more, of the following destinations: (1) patient record (default), (2) printer nearest the patient, (3) pharmacy, and (4) personal file of the creator of the sector.

HCOM has the ability to ask for data (the DDA data-acquisition mode referred to above) that is not found in the patient's file. The operator must specify who is to be asked (patient, doctor, or nurse), whether the data is to be requested hierarchically (ie, ask for cough before asking for cough with sputum), and which search items (ie, A,D,N) are needed.

The LE (list and explain) command presents each arithmetic or probabilistic statement with the details of the search items it uses. Thus, the user can read the meaning of each item whose label is represented in the function. A debug mode allows the user to run the sector against any patient's data and see each item displayed with the resulting value, exist, and time parameters.

PAL—the Report Generation Software Tool

Creation of reports is one of the most essential parts of the HELP system. The HELP data base contained no commercial report generation language for creating patient-oriented reports from data stored in the system. To allow for efficient report development and execution, a report generation language, called PAL was designed to allow interaction with the HELP data base. Two major concerns were addressed when designing PAL: (1) ease of access to the HELP data base; (2) ease of report formatting and distribution.

```
RELATION penicillin;
BEGIN
ITEM time                                STRINGTIME;
ITEM drug                  CODE(8 1 2 2 10*);
END;
```

Figure 21.2. A relation to retrieve all forms of penicillin ordered on a patient.

To facilitate access to the HELP data base, two key constructs were designed into PAL. The first converted the HELP hierarchical data base into a local relational data base within PAL. The local relation model was implemented because of its simplicity in dealing with small data bases. This was implemented by defining a RELATION construct in PAL. Using this construct, the programmer defined a relationship consisting of data elements from the same field code defined in the PTXT data dictionary. For example, a complete blood count (cbc) relation might consist of the white blood cell count, red blood cell count, hemoglobin (hgb), and hematocrit (hct). Since those elements had previously been defined in PTXT under the same data class and field code, a relation in PAL could be created that both optimizes the disc access of the data and groups the retrieved data according to time-associated sets(tuples). Figure 21.1 illustrates the PAL syntax necessary to create this relation in a PAL program. The codes within parenthesis correspond to the PTXT hierarchical codes. This example shows a simple relation where the data elements are completely specified. However, because of the hierarchical nature of the HELP data base, it is often necessary to define a data element (variable in PAL) as any which satisfies the constraints of a certain portion of the hierarchical code. For example, if a report is being created to list the drugs given a patient, the relation would be defined to capture all drugs of a certain PTXT level. The example in Figure 21.2 is such a relation. Here the PTXT code, specifying the drug, is defined as a specific noun, while allowing any adjective under that noun to satisfy the search condition. This allows all strings in the patient's record which are of the specified data class, field code, and noun to be retrieved and stored into the defined relation. References to data elements may also be defined individually and not as a part of a relation. The use of the relation construction, however, causes relational tables to be built. When defining a relation, one of the variables that is usually defined is the time of each tuple (set of variables for one row of the relational table). In Figure 21.1 the variable CBC time contains the time associated with each tuple of the relation.

The second construct, which facilitates access to the data base, is the BUILD statement. This statement actually performs the search and retrieval of data from the patient's file. The experience gained from the SEARCH statement of HCOM was instrumental in designing the syntax of the BUILD statement. The basic syntax for the BUILD statement is BUILD <relation/item expression> WHERE<where clause> MAXCOUNT <number><time frame>. This statement causes the requested items/relations to be retrieved and stored in the

appropriate space defined by the RELATION or ITEM definitions. The WHERE clause retrieves only the patient data strings satisfying the logic (values, time, etc.) The MAXCOUNT clause is used to limit the number of strings retrieved within a given time frame. The syntax for specifying a time frame is FROM <early time> TO <late time> for a forward search and FROM <late time> BACKTO <early time> for a backward search. The following example illustrates these features: BUILD cbc WHERE cbc.hgb < 12 or cbc.hct <38 MAXCOUNT 10 FROM $NOW BACKTO $NOW - 5 DAYS. In this example, the BUILD statement searches the patient's data and creates a relation previously defined as cbc. The patient strings, which are included in the relation, are those cbc strings where the hgb is less than 12 or the hct is less than 38. The MAXCOUNT clause restricts the size of the relation to ten tuples and terminates the search when ten valid strings have been retrieved. The time constraints listed restrict the search; it begins with the latest string in the patient file and searches backwards in time to five days before the time of the search request. The power of the BUILD statement is increased, since several items/relations may be retrieved in a single BUILD command. This capability has resulted in important improvements in run-time efficiency, since optimal search requests are easily constructed from a single BUILD command.

The formatting and routing of reports to users (terminals and printers) are activated through specific constructs incorporated into PAL. The first is a WRITE statement, which is used to format and send messages to either a file (location on the computer including disc, terminal, or printer) or a character string variable. The syntax of the WRITE statement is WRITE <file expression> <variable list> FORMAT <format statement> where file expression designates the file to where the message is to be written. If no file is specified the message is sent to a default device. The default device is usually the terminal or the printer nearest the terminal where the program is being run. The variable list is an optional list of ITEMs, RELATIONs, and VARIABLEs. The format statement is a FORTRAN-like statement specifying the format of the message. Special formats used with the HELP system include the *P format, which causes the text defined in the HELP data dictionary to be placed in the formatted message, and the T format, which prints the time of the item. Additional special format items allow control of the screen attributes of the computer terminal. These attributes include screen erase, half intensity, blink, etc. The following is an example of a simple WRITE statement in PAL: WRITEcbc.time,cbc.hgb,cbc.hctFORMAT(T(DAY,MONTH,"",HR,":",MIN),P10,"Hematocrit",F6.2) This statement causes a message to be sent to the terminal or printer displaying the hemoglobin level and the hematocrit level and the time complete blood count data was stored in the patient file. The text printed before the hemoglobin value will be the text stored in the HELP data dictionary, whereas the text printed before the hematocrit will be "Hematocrit." Since the data dictionary specifies the format of the data item, no value format is required when using the P format type. If, however, the following construct message had been used, the formatted message would be entered into the string variable "message" and not transmitted

```
SECTION SMA7;
BEGIN
VARIABLE I;

RELATION SMA7'
BEGIN
ITEM TIME                                        STRINGTIME;
ITEM SPEC^TIME                          CODE(13 1 0 4 0 0 0 1);
ITEM NACODE(13 1 1 1 1);
ITEM KCODE(13 1 1 1 2) SCALE(-1);
ITEM CLCODE(13 1 1 1 3);
ITEM COCODE(13 1 1 1 4);
ITEM GLCODE(13 1 1 1 5);
ITEM CRCODE(13 1 1 1 6);
ITEM BNCODE(13 1 1 1 7);
ITEM COMMENT                            CODE(13 1 1 4 0 0 0 2);
ITEM FUTR^CMT                                 CODE(13 1 1 4 0 0 0 6);
END;

BUILD SMA7 FROM $LATE^TIME BACKTO $EARLY^TIME;
WRITE FORMAT(2/,"LAB DATA - SMA-7",2/,      "DATA   TIME   NA+
  K+     CL-     CO2     BUN     GLUC   CREAT");

FOR I: = 1 TO $COUNT(SMA7)DO
BEGIN
WRITE SMA7[I].TIME,SMA7[I].SPEC^TYPE,SMA7[I].NA,
  SMA7[I].KSMA7[I].CLSMA7[I].CO,SMA7[I].BN,
  SMA7[I].GL,SMA7[I].CR
FORMAT(C1,T(DAY,MONTH," ",HR,":"MIN),2X,A2,F7,F7.1,
  4F7,F7.1);
IF $EXIST(SMA7[I].COMMENT)
  THEN WRITE SMA7[I].COMMENT FORMAT(C14,P80);
IF $EXIST(SMA7[I].FUTR^CMT)

  THEN WRITE SMA7[I].FUTR^CMT FORMAT(C14,P80);
  END;
END;
```

Figure 21.3. A PAL program to creat a Sequential Multiple Analysis-7 (SMA-7) report.

to any file. :=WRITE cbc.time,cbc.hgb,cbc.hctFORMAT(T(DAY,MONTH," ",HR,":",MIN),P10,"Hematocrit", F6.2) This use of the WRITE statement must be used in conjunction with the DISPATCH statement.

The DISPATCH statement is used to route messages to various terminals, or printers, throughout the hospital. This statement uses the string variable created by a WRITE statement (the variable "message" in the example above) and the reserved variable $LOCATION. The $LOCATION variable can designate specific locations (such as the printer in the ECG laboratory) or generic locations (such as the printer nearest the patient's room, or the printer nearest the terminal where the program is being executed). The $LOCATION can contain more than one location, thus allowing a report to be transmitted to several places simultaneously. For example, a blood gas order can be displayed on the terminal where the order is created, at the printer nearby and simultaneously at the printer in the blood gas laboratory.

Patient, Test 1234567 E715

LAB DATA-SMA-7

DATA TIME		NA+	K+	CL−	CO2	BUN	GLUC	CREAT
04SEP 04:40	B	132	4.1	106	23	39	150	1.2
03SEP 18:35	B	3.8						
03SEP 04:00	B	132	4.4	104	24	36	159	1.2
02SEP 22:46	B	3.7						
02SEP 04:29	B	131	3.7	107	24	30	176	1.3
01SEP 04:50	B	130	4.1	97	26	24	97	1.2
31AUG 22:25	B	127	4.6	98	25	21	106	1.2

COMMENT: SEE PRINTED LAB REPORT FOR COMMENTS

Figure 21.4. An example of an SMA-7 report created by the PAL program of Figure 21.3.

The special constructs used by the HELP system, together with general structured programming statements such as WHILE or FOR statements, along with the math library, have made PAL an extremely powerful report-generating tool. Because of the versatility of PAL, many other hospital applications have recently incorporated it rather than the Tandem Application Language (TAL) previously used. Figure 21.3 is the PAL program written to create the report in Figure 21.4. This report retrieves a patient's SMA-7 (Sequential Multiple Analysis-7) results and formats them in a tabular form with the most recent report displayed first.

Much of the efficiency and flexibility of PAL derives from the object code created by the PAL compiler. The compiled output of the PAL programs is not executable code, but a set of Pcodes (pseudocodes). These Pcodes are then interpreted by a special interpreter written for the HELP system. By using low-level Pcodes for execution of algebraic statements, and high-level Pcodes that link to the HELP library routines, a level of efficiency that meets the throughput requirement needs for an on-line hospital information system has been gained.

STRATO—the Research Subsystem

A package called STRATO was designed, in connection with the HELP system, to support medical research. It derives its name from the first step in data analysis, the stratification of the larger patient data base into subgroups for data analysis.

The central purpose of the STRATO program is to identify, in a clinical data base, those patients meeting specific criteria and to extract the information required by the researcher. The HELP data base is designed, however, for review of patient-specific data, not group-specific data. The daily clinical use, for which the HELP data base was designed, stresses rapid, flexible access to types of information from a single patient record. Research goals dictate accessing a restricted set of information from a large number of patient records. STRATO's approach is to create lists of patients meeting specified research criteria, and then to assemble the required data points for each patient (from search criteria created by the research user).

STRATO has four methods for generating the initial patient list. The first is to search the clinical data base for patients with specific criteria. Patients are selected by their patient number, name, the rooms they have been in, admit and discharge dates, attending physicians, or by which service they are on. This mode of patient selection is based on information in the patient's identification (ID) file and results in rapid searches.

The second method of searching involves choosing criteria for selection based on the clinical data stored in the computerized patient record. All the data collected by the HELP system is accessible to this type of search. Patients can be selected by laboratory results, discharge diagnoses, medications taken during hospitalization, or by a wide variety of other criteria. The searches can be restricted according to the existence of the data sought, the values of the data sought, or the times when the data were acquired. A group of modifiers (FIRST, LAST, ANY, FREQ, ORIGINAL, MIN, MAX), similar to those provided in HCOM, allow further specification of the data requirements. A major advantage of this strategy is its ability to restrict searches to specific time intervals. As an example, it might be valuable to find all patients with a creatinine level greater than 2.0 mg/dL measured three to five days after therapy with gentamicin was begun. The time interval is based on the earliest dose of gentamicin and is different for each patient.

Search criteria are constructed using an interactive, menu-driven command language resident in the STRATO program. As the user constructs search criteria, a process called ENTR mediates access to the HELP data dictionary. Keywords are used to find specific data entries in the dictionary. Mathematical and logical combinations of these data can then be specified and further restrictions based on time, or the modifiers listed above, can be entered. Finally, the patient population is selected. This can be either a complete patient data base containing up to six months of inpatient and outpatient admissions (approx. 35,000 patients), or a subset of such a data base defined in a previous search.

The third approach to patient selection is to use the HELP decision frames to specify criteria. To do this, a frame is created in the HELP decision language. This frame defines the characteristics of the required patients. Within a frame, a wide range of data can be simultaneously accessed and manipulated using a variety of logical and mathematical operators and functions. This approach is equivalent to writing a small program to describe the search patients and is best

used for complex searches (eg, finding all those patients in a data base whose gentamicin elimination kinetics required a reduction in gentamicin dose). In this case, the HELP frame would have to access gentamicin doses, gentamicin levels, and the times of these events, and compute the pharmacodynamic characteristics of gentamicin in each patient before determining whether to include him in a chosen group.

Finally, options are available in STRATO for combining previously created patient groups. New patient lists can be formed from old ones through the familiar operations of union, intersection, and formation of the compliment of the intersection.

The result of each search is a "population," a list of patients that match the specified criteria. A series of searches can further divide this list into subpopulations separated by additional criteria. Research populations, with chosen characteristics, are created using lists of pointers to the original clinical data base.

Patient data are extracted from the clinical data base in a similar way. The same tools that select search criteria are used to indicate which data are sought from the data base. Similar use is made of keywords, mathematical and logical combinations, modifiers, and time restrictions. Once the data have been specified, the researcher is asked to choose a population from which to retrieve the data. Typically, he indicates one of the population lists created during the patient selection process. The results of this search are returned as a list of data elements. These have a one-to-one correspondence with the patient list from which they were generated. Where data meeting criteria cannot be found, a flag is inserted in the data list to indicate that the required patient information was missing.

These lists of data are called "variables." Up to 250 variables can be associated with any patient population. As in patient selection, complex, derived data can be specified through the use of HELP decision frames.

The STRATO system can also be used to test HELP decision frames. As the frames are created, and before they become clinically active, the accuracy of their logic can be verified with STRATO.

In a large clinical data base like the HELP system, most combinations of data that might challenge the decision logic are represented. This allows thorough testing of new decisions based on actual patient data. With STRATO, decision frames can be run against varied patient populations.

Since the decisions made by the system are stored in the computerized patient record, STRATO can be used for reviewing the decision frames by searching for all patients who receive a given decision made by the system. The decisions are then compared with corresponding ones made by clinicians. Discrepancies are evaluated and lead to improvements of the decision logic.

The STRATO program is primarily concerned with the selection of patients meeting certain criteria and with the extraction of data from their records for later analysis. Sophisticated statistical testing procedures are not provided within the STRATO program. Instead, a variety of programs are available within the HELP system that accept STRATO files and allow further analysis. These include routines for analysis of variance, multivariate regression, chi-square, Wilcoxen

rank sum, and others. Processes supporting graphical representation of STRATO data are also provided.

However, with the availability of flexible data bases and good statistical software on personal computers, we now provide tools for exporting the files created by STRATO to personal computers. This has the advantage of removing statistical processing from the HELP system hardware and freeing resources for development and support of clinical service. We currently support procedures for exporting the files created by STRATO to both IBM and its compatibles and Macintosh microcomputers.

We have gone a step further with the Macintosh by developing software that allows the entire STRATO data-collection routine to be mediated through the user-friendly interface tools that are a part of the operating system. Once the population and variable search criteria are defined within the Macintosh, they are submitted to STRATO and processed. The Macintosh supervises the transfer of the collected data back into its file system and formats that data for direct access by statistical and spreadsheet packages. To ensure error-free transmission data during transfer, a KERMIT protocol is used on each system. Similar interfaces are under development for the IBM personal computer.

Downloading makes it possible to generate personal data bases for those who would like to do clinical research without the complications imposed by a busy hospital information system, as well as having access to a variety of standard, personal-computer-based, data analysis and graphics programs.

Development is progressing on each of the HELP system software tools presented in this chapter. The goals of this development are to increase the integration of those subsystems in order that the applications developer may, upon learning a single application language, more easily create and evaluate clinical and research applications on the HELP system. Increased capability will enhance the decision-support logic, making that technology available to new classes of hospital information problems.

Bibliography

1. Pryor TA, Gardner RM, Clayton PD, et al: The HELP system. J Med Syst 1983;7:87–102.
2. Cengiz M, Ranzenberger J, Johnson DS, et al: Design and implementation of computerized nursing care plans. SCAMC 1983;7:561–565.
3. Pryor TA, Gardner RM, Clayton PD, et al: The HELP system, in Blum BI (ed): Patient Information Systems for Patient Care. New York, Springer-Verlag, 1984, pp 109–128.
4. Bradshaw KE, Gardner RM, Clemmer TP, et al: Physician decision-making–evaluation of data used in a computerized ICU. Int J Clin Monitor Comp 1984;1:81–91.
5. Gardner RM: Tomorrow's electronic hospital is here today. IEEE Spectrum June, 1984;21:101–103.
6. Gardner RM, Pryor TA, Clayton PD, et al: Integrated computer network for acute patient care. SCAMC 1984;8:185–188.

7. Pryor DB, Barnett GO, Gardner RM, et al: Measuring the value of information systems. SCAMC 1984;8:26–28.
8. White KS, Lindsay A, Pryor TA, et al: Application of a computerized medical decision-making process to the problem of Digoxin intoxication. J Am Coll Cardiol 1984;4:571–576.
9. Chapman RH, Ranzenberger J, Pryor TA: Computerized charting at the bedside: promoting the nursing process. SCAMC 1984;8:700–702.
10. Killpack AK, Johnson DS, Pryor TA, et al: Automating patient acuity from nursing documentation. SCAMC 1984;8:709–711.
11. Pryor TA, Goldberg RD, Brown WF, et al: Computerized management of arrythmias. Comput Cardiol 1984;11:39–44.
12. Liddle HV, Gould BL, Jones PD, et al: Conditional probability of multiple coronary graft failure. J Thorac Cardiovasc Surg 1984;87:526–531.
13. Gould BL, Clayton PD, Jensen RL, et al: Association between early graft patency and late outcome for coronary artery bypass graft patency. Circulation 1984;69:569–576.
14. Clayton PD, Haug PJ, Gerard MJ, et al: The role of radiology findings in automated decision making. Proc 8th ACR Conf on Computer Applications in Radiology, St. Louis, May 1984, pp 521–531.
15. Ostler DV, Gardner RM, Crapo RO: A computer system for analysis and transmission of spirometry waveforms using volume sampling. Comput Biomed Res 1984;17:229–240.
16. Evans RS, Gardner RM, Bush AR, et al: Development of a computerized infectious disease monitor (CIDM). Comput Biomed 1985;18:103–113.
17. Gardner RM: Computerized data management and decision-making in critical care. Surg Clin North Am 1985;65(4):1041–1051.
18. Andrews RD, Gardner RM, Metcalf SM, et al: Computer charting: An evaluation of a respiratory care computer system. Respir Care 1985;30:695–707.
19. Haug PJ, Warner HR: Physician-oriented applications of artificial intelligence, in Clinics in Computer Applications in Medicine. WB Saunders 1988, (to be published).
20. Ostler MR, Stansfield JD, Pryor TA: A new, efficient version of HELP. SCAMC 1985;9:269–297.
21. Bekemeyer WB, Calhoon S, Crapo RO, et al: Efficacy of chest radiography in a respiratory intensive care unit–a prospective study. 1985;88:691–696.
22. Gardner RM: Editorial–Artificial intelligence in medicine–Is it ready? Int J Clin Monitor Comput 1986;2:133–134.
23. Evans RS, Larsen RA, Burke JP, et al: Computer surveillance of hospital-acquired infections and antibiotic use. JAMA 1986;256:1007–1011.
24. Giles DJ, Thomas RJ, Osborn AG, et al: Lumbarspine: Pretest predictability of CT findings. Radiology 1984;150:719–722.
25. Gerard MJ, Haug PJ, Morrison WJ, et al: A computer system for diagnosing pulmonary artery disease. Proc Am Assoc Med Inf, San Francisco, May, 1984, pp 119–123.
26. Clayton PD, Pryor TA, Gardner RM, et al: HELP: A Medical Information System with Decision Making Capability. 6th International Congress of Medical Informatics. Roger FH, Gronroos P, Tervo-Pellikka R, et al (eds): Helsinki, Finald. Heidelberg, Springer-Verlag, 1985, pp 127–131.